INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

A low latency ASR-free end to end spoken language understanding system

Mohamed Mhiri, Samuel Myer, Vikrant Singh Tomar

Fluent.ai Inc., Montréal, Québec, Canada
mohamed.mhiri@fluent.ai,sam.myer@fluent.ai,vikrant@fluent.ai

Abstract

In recent years, developing a speech understanding system that classifies a waveform to structured data, such as intents and slots, without first transcribing the speech to text has emerged as an interesting research problem. This work proposes such as system with an additional constraint of designing a system that has a small enough footprint to run on small micro-controllers and embedded systems with minimal latency. Given a streaming input speech signal, the proposed system can process it segment-by-segment without the need to have the entire stream at the moment of processing. The proposed system is evaluated on the publicly available Fluent Speech Commands dataset. Experiments show that the proposed system yields state-of-the-art performance with the advantage of low latency and a much smaller model when compared to other published works on the same task.

1. Introduction

Nowadays, spoken language understanding (SLU) systems are crucial for daily life communication, where they can provide the crucial vocal user interface to home-controller devices and other appliances. The role of an SLU system is to convert a given speech signal to a structured representation, such as in the form of intent/slots classes, that could be interpreted by a software and application to ultimately perform an action on the target device [1, 2]. For example, a speech signal like 'set an alarm for 5 p.m.' might have the following representation {intent: 'SET_ALARM', data_time: '5 P.M.'}.

In classical SLU systems, an automatic speech recognition (ASR) model is first used to transcribe speech signals to a word string, followed by a natural language understanding (NLU) model that classifies this word string to the target intent representation. While this approach works well in various scenarios, there are a number of problems. For instance, as mentioned in [3, 4], the two models, ASR and NLU, are often trained independently and not jointly optimized. Most ASR systems in themselves consists of a number of dis-jointly trained components. These issues can affect the overall performance of the SLU systems. Furthermore, these models often have high data and computational requirements, limiting their applicability to a handful of use-cases and languages.

The number of recent end-to-end and ASR-free SLU systems show promising results [5–8]. These SLU systems map a speech signal directly to the speaker's intent without explicitly recognizing the corresponding text. As with the conventional ASR+NLU systems, the end to end SLU systems are often computationally demanding. For example, the SLU systems presented in [5–8] give good performance in terms of the recognition accuracy, however, they are a combination of several neural networks with each having hundreds of millions of parameters. Running such systems in real-time on low-power devices is not feasible.

In this work, we present an efficient and compact end-to-end SLU system. The proposed system is targeted at low-footprint devices, where the entire speech data is processed on the device without the need to send any data to a cloud server. By processing the entire speech data on the device, such a system provides increased privacy for the user. Furthermore, such a system can enable voice user interfaces for a number of use-cases and applications that would not have been possible in an always-connected scenario. There are a number of technical contributions in this work that provide the aforementioned advantages. These are summarized below.

- In order to have the low latency property, the proposed model process a given input speech signal segment-by-segment. Here, the processing of one segment is done while receiving the upcoming one. The processing begins even if the speech signal is not entirely received.
- The proposed model is built using convolutional layers, which are less computationally expensive than recurrent layers [9–11].
- The proposed model can process speech signals of variable duration without requiring any padding or cutting of the incoming speech.

2. Related work

In this section, we present the current state-of-the-art of the end-to-end SLU systems.

In [3], the proposed SLU system is a speech-to-intent approach tested for semantic classification in dialog systems. In this system, the given speech signals are mapped directly to semantic meaning. This has several advantages. First, richer information than words can be extracted from speech. Second, the ability to extract semantic meaning from mixed language speech. The model proposed in [3], is composed of an acoustic model pre-trained with CTC loss to predict graphemes, and a semantic model pre-trained with the outputs of the acoustic model to predict intents. Once the two models are pre-trained, a full pipeline training (i.e., a fine tuning) is done on the entire architecture. In [6], a similar framework to [3] is proposed, which is composed of three models. The three models are pre-trained respectively with phonemes, words, and intents. Then, they are fine tuned by training them together.

As opposed to [3, 6], in [8] no pre-training is used, which is similar to our model. Here in [8], an encoder-decoder framework is proposed. The encoder is a multi-layer bidirectional recurrent layers. The decoder maps the output of the encoder to its corresponding intent class. To reduce the computational time, Serdyuk et al. used a sub-sampling for the Hidden activations along the time domain [12, 13].

Recently, Haghaniet et al. [5] proposed and compared four different SLU encoder-decoder based approaches, which are augmented by the attention mechanism [12]. In all these proposed approaches, the mapping of speech signals to intents is

http://dx.doi.org/10.21437/Interspeech.2020-1449

formulated as a sequence-to-sequence problem [14, 15]. The first approach maps audio features directly to their corresponding semantic sequence (domain/intent/arguments). In the second approach, the decoder outputs, for a given speech signal, not only its corresponding semantic sequence, but also its sequence of graphemes. The third has two decoder models: one outputs its semantic sequence and another outputs its sequence of graphemes. In the last one, called a multistage model, two stages are used where, in the first stage, the transcript is predicted, and in the second stage, the semantics are predicted. Here, the two stages are independently optimized and afterwards the whole system is fine tuned together. Haghaniet et al. [5] conclude, after evaluations on real-world scenarios, that having an intermediate text representation and jointly optimizing the full system improves the overall accuracy of prediction.

In all these approaches [3, 5, 6, 8], the proposed network architectures are heavily based on recurrent layers, which may not be well suited for low-power devices [16]. These types of layers are slower and less computationally efficient than convolutional layers [9–11].

3. Proposed Approach

This section presents the proposed architecture along with problem formulation and the relevant implementation details.

3.1. Problem formulation

The speech-to-intent model proposed in this work is composed of a sequence of convolutional layers followed by a global max-pooling layer and few fully connected layers that output the intent class (see Table 1). Here, the global max-pooling layer has crucial importance since it allows processing any given input speech signal segment-by-segment without the need to have it entirely at the moment of processing. In addition, this layer allows to process speech signals with different variable length, where no padding or concatenating is demanded.

In Figure 1, we show the difference between processing a full speech signal versus segment-by-segment speech signal processing (i.e., the proposed scenario). As is shown, in the full-signal scenario, we wait till receiving the full speech signal then we process it (i.e., forwarded it through the model). However, in segment-by-segment signal processing scenario, each segment is forwarded separately through the convolutional layers and the global max-pooling layer. Then, all the outputs of all segments are stacked together and we again apply the max-pooling through them and we forward the resulting vector through the fully connected layers to classify the intent. Here, the advantage is that we can process one segment, while we are receiving the upcoming one. This results in less processing time, since a large part of the processing is done while receiving the speech signal.

For example, let $\{I_0, \ldots, I_T\}$ be the input acoustic features sequence for a given speech signal.

- In the full-signal processing scenario, the convolutional layers output a sequence $\{f_0, \ldots, f_N\}$ then the global max-pooling layer pool this sequence of outputs to one vector R, which is mapped by the fully connected layers to the intent class.
- In the segment-by-segment speech signal scenario (i.e., Algorithm 1), the input acoustic features sequence can be viewed as two segments $\{I_0, \ldots, I_T\} = \{I_0, \ldots, I_s\} \cup \{I_{s+1}, \ldots, I_T\}$, (i.e., in this example, there is no overlapping between segments). The convolutional layers output two sequence $\{f_0, \ldots, f_n\}$ and $\{f_0, \ldots, f_m\}$ for the two segments. Then, a global max-pooling layer pool the two sequences of outputs to two vector R_1 and R_2. In the last step, we apply again the global max-pooling for $\{R_1, R_2\}$ and outputs one vector R, which is mapped through the fully connected layers to the intent class.

Algorithm 1: The segment-by-segment speech signal processing algorithm.

Input: A stream input speech signals.
Output: The intent class.

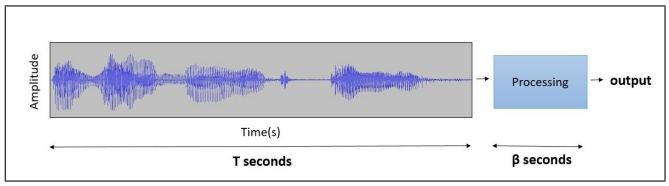
```
T=[];
while waiting the upcoming segment (S_{i+1}) do
    R_i=global-max-pooling(Conv-layers(S_i));
    T=[T,R_i];
end
R=global-max-pooling(T);
output=fully-connected(R);
return output
```

The segment by segment processing is only used in the inference time. However, in the training time, the complete signal is processed at once. In the full-signal processing scenario, we need a total time of $T + \beta$ (seconds), where T is the time to receive the speech signal and β is the processing time. However, in the segment-by-segment speech signal scenario, the proposed approach needs less processing time. Since a part of the processing is done when receiving the speech signal. Here, the exact processing time depends on the two hyper-parameters: segment size and step size; these represent the size of the segment to process it each time and the size between the beginnings of two consecutive segments, respectively.

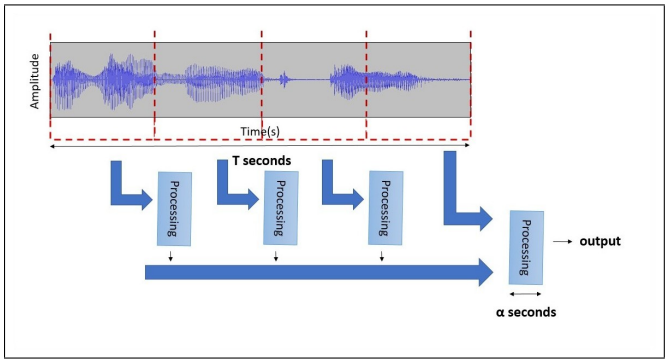
3.2. The proposed network architecture

Table 1 presents details about the proposed network architecture. Here, the global max-pooling layer is a critical component. It allows processing speech signals of variable duration segment-by-segment. The proposed network is composed of 17 layers in total, with 8 convolutional layers, 4 max-pooling layers, 1 global max-pooling layer, and 4 fully-connected layers. All hidden layers use rectified linear units (ReLU) [17]. Moreover, as recommended in [18], batch normalization is applied before each activation layer. The 8 convolutional layers with the 4 max-pooling layers can be viewed as 4 blocks. Where, each block is composed of one convolutional layer followed by a max-pooling layer of (2×1) kernel and a convolutional layer of (1×1) kernel. The (1×1) convolutional layer is used mainly to reduce the number of features and to keep a small-footprint model.

In Table 1, an input of size (100×41) is used. This input represents one second of speech, where the acoustic features are extracted each 10ms for a speech signal frame sized of 25ms. The 41 elements are the 40 filters banks and the energy measure for the corresponding frame. As mentioned in [19], applying the cepstral mean and variance normalization (CMVN) on these 41 features improves speech recognition/classification performances. This normalization reduces the environmental changes and mismatch between the training and the testing conditions, where different background noises or different microphones can be presented. In full-signal processing scenario, the CMVN is usually applied at the utterance level. However, in



a) Full-signal processing scenario.



b) Segment-by-segment signal processing scenario (the proposed approach).

Figure 1: *A comparison between traditional approaches that encode speech signal after receiving it fully and the proposed approach that encodes it segment-by-segment while receiving it. The first approach needs a total time $\boldsymbol{T} + \boldsymbol{\beta}$ (seconds) (T is the long of the speech signal), while the proposed approach needs a $\boldsymbol{T} + \boldsymbol{\alpha}$ (seconds). Here, $\boldsymbol{\beta} > \boldsymbol{\alpha}$ since a part of the processing is done when receiving the speech signal, which gives a low-latency SLU system.*

Table 1: *The CNN architecture for speech segment of size* 100×41 *(1 second).*

Layer type	**Output shape**
4×41, Conv2D, 128	$128 \times 97 \times 1$
Max-pooling	$128 \times 48 \times 1$
1×1, Conv2D, 64	$64 \times 48 \times 1$
4×1, Conv2D, 128	$128 \times 45 \times 1$
Max-pooling	$128 \times 22 \times 1$
1×1, Conv2D, 64	$64 \times 22 \times 1$
4×1, Conv2D, 128	$128 \times 19 \times 1$
Max-pooling	$128 \times 9 \times 1$
1×1, Conv2D, 64	$64 \times 9 \times 1$
4×1, Conv2D, 256	$256 \times 6 \times 1$
Max-pooling	$256 \times 3 \times 1$
1×1, Conv2D, 256	$256 \times 3 \times 1$
Global max pooling	**256**
Fully-connected (ReLU units)	256
Fully-connected (ReLU units)	196
Fully-connected (ReLU units)	128
Fully-connected (Softmax)	31*

* *Number of intents in the Fluent Speech Commands dataset.*

the proposed segment-by-segment speech signal scenario, utterance CMVN can not be applied because it requires having received the whole speech signal in the moment of processing. Instead, we use globally computed mean and variance with all the training data [20, 21].

4. Results

Table 2: *Information about the Fluent Speech Commands dataset.*

Split	**# of speakers**	**# of utterances**	**# hours**
Train	77	23,132	14.7
Validation	10	3,118	1.9
Test	10	3,793	2.4
Total	97	30.043	19.0

This section summaries the experiments conducted in this work. First, we present the performance in terms of recognition accuracy of the proposed approach on the publicly available Fluent Speech Commands dataset [6]. Table 2 summarizes some information about the dataset. Next, we evaluated the impact of different CMVN optimizations on the recognition accuracy. Finally, we evaluate the impact of various hyperparameters on the proposed approach.

4.1. Comparison to state-of-the-art

Table 3: *Error rate on the clean testing set comparing to state-of-the-art models using the Fluent Speech Commands dataset.*

Method	**clean**	**Model size**
Lugosch et al. (Full pipeline) [6]	1.2%	14.6M
Lugosch et al. (No pre-training) [6]	3.6%	14.6M
Poncelet et al. (Capsule net) [22]	1.9%	-
Palogiannidi et al. [23]	1.38%-5.83%	-
Proposed model	2.18%	**1.3M**

Table 3 presents the results in terms of command recognition accuracy on the clean Fluent Speech Commands test set for the proposed approach along with other works from recent research. Lugosch et al. in [6] used a model composed of a hierarchy of sub-networks. The first sub-network is trained to recognize phonemes from acoustic features. The second one is trained to recognize words from phonemes. The last one is trained to recognize intents from words. In the full pipeline scenario, Lugosch et al. first pre-trained the three sub-networks separately then the entire model is fine-tuned. Our approach can be compared to the approach of Lugosch et al. [6], where no pre-trained model is used. We can observer from the table that our approach has better performance (+1.42%). In addition, our model is only $1.3MB$ in size compared to $14.6MB$ for Lugosch's model. The work of Poncelet el al. [22] has a comparable performance to ours (1.9% vs 2.18%) similar to the work of Palogiannidi et al. [23], which has performances between 1.38% and 5.83% depending on the number and the types of the recurrent layers. Our approach is more appropriate for low-

power devices since it is built from only convolutional layers, where the models of [6, 22, 23] includes many recurrent layers, which are slower compared to convolutional layers [9–11].

4.2. Comparison of using different CMVN normalization

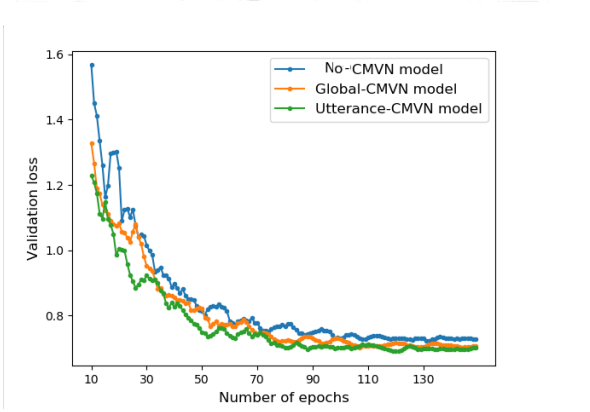


Figure 2: *Validation loss for no CMVN, Global CMVN and Utterance CMVN models.*

Figure 2 shows the validation losses for the three models trained respectively, with global CMVN, utterance CMVN and without any CMVN (No-CMVN). We can see that applying the global CMVN and utterance CMVN has almost the same performance. However, in the segment-by-segment processing scenario, we cannot apply utterance based CMVN, therefore, our only options are either applying the global CMVN or no CMVN at all. It is evident from the results in the Table 4 that applying the global CMVN leads to a higher performance. In the table, the column labeled '5dB' represents the testing case when the Fluent dataset is enhanced by augmenting with a mix noise types with a signal-to-noise ratio of 5dB. The '5dB + ff' scenario refers to the '5dB' set that is further augmented by a set of real Room Impulse Responses (RIRs). For the rest of this paper, all the experiments are performed with the global CMVN model. This is also true for the results presented for the proposed algorithm in Table 3.

Table 4: *Error rate on the testing set for no CMVN, Global CMVN and Utterance CMVN models.*

Method	**clean**	**5dB**	**5dB + ff**
no CMVN	2.18%	10.23%	19.11%
Global CMVN	**2.18%**	**9.96%**	**18.98%**
Utterance CMVN	2.48%	9.1%	18.4%

4.3. Hyper-parameters analysis

There are two main hyper-parameters in the segment-by-segment signal processing scenario. The first is the segment size, S, that represents the size of the audio segment to process. The second is the step size, T, that represents the amount which the window is moved between two consecutive segments. This means that each T seconds, we process the last S seconds till the ending of the speech signal. When, the step size is less than the segment size, this means that there is an overlapping between the segments.

Table 5: *Mean Error rate on the (clean, 5dB and 5dB+ff) testing set for the segment-by-segment signal processing scenario using different hyper-parameters values.*

(Step $T\downarrow$ / Segment $S\rightarrow$)	**1s**	**1.25s**	**1.5s**	**1.75s**	**2s**
0.25s	**10.47%**	10.61%	10.81%	10.81%	10.8%
0.5s	10.57%	10.64%	10.54%	**10.51%**	10.53%
0.75s	24.38%	10.81%	10.47%	**10.26%**	11.07%
1s	46.63%	17.53%	11.43%	10.54%	**10.53%**
1.25s	61.49%	34.67%	20.85%	**17.43%**	13.95%
1.5s	69.42%	49.56%	38%	15.93%	**11.49%**

Table 5 shows that smaller step size, and hence the overlapping between segments, is crucial. A small step size means more information to encode, while information can be seen in different segments. This results in a better and richer representation. On the other hand, a higher step size means less computation. However, since a segment is processed while the system is receiving the next segment, having a high step size is not beneficial. Furthermore, we can observe that using a high segment size is not always beneficial. A small segment size also is not preferred since it can effect the overlapping. To conclude, the best hyper-parameters combination is the one that preserve a higher overlapping between segments.

Table 6: *Mean Error rate on the (clean, 5dB and 5dB+ff) testing set for the segment-by-segment signal processing scenario versus the full-signal processing scenario.*

hyper-parameters	**Mean Error rate**	**Time of processing needed after fully receiving the speech signal (%)**
(Segment, Step)=(1.75s, 0.75s)	**10.26%**	43%
(Segment, Step)=(1s, 0.25s)	10.47%	**25%**
Full signal speech	10.37%	100%

Table 6 shows the difference in performances between the segment-by-segment signal processing scenario and the full signal processing scenario. We can see that the segment-by-segment processing not only reduces the time of processing but also improves the performance (10.26% vs 10.37%). Here, the time of processing needed after fully receiving the speech signal is represented as a percentage. In full signal processing scenario (Figure 1), it is represented by (β seconds) equal to 100%. In segment-by-segment processing scenario with the hyper-parameters (Segment, Step)=(1.75s, 0.75s), the time of processing is 43% of β. This means that after fully receiving the speech signal, we wait 2.3 times less with the segment-by-segment processing scenario than the full signal processing scenario.

5. Conclusions

In this paper, we presented a speech-to-intent model for low-power devices. The proposed model has a global max-pooling layer that allows not only processing no fixed-length speech signals. But also, it allows processing any speech signal without the need to have it entirely at the moment of processing. The proposed processing scenario is done segment-by-segment, which means that the processing of one segment is done while we are receiving the upcoming segment. This not only reduces the response time but also improves the performances.

6. References

[1] G. Tur, *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*. John Wiley and Sons, January 2011.

[2] A. Bapna, G. Tür, D. Hakkani-Tür, and L. Heck, "Sequential dialogue context modeling for spoken language understanding," in *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*. Saarbrücken, Germany: Association for Computational Linguistics, Aug. 2017, pp. 103–114.

[3] Y. Chen, R. Price, and S. Bangalore, "Spoken language understanding without speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018, pp. 6189–6193.

[4] S. Ghannay, A. Caubrière, Y. Estève, N. Camelin, E. Simonnet, A. Laurent, and E. Morin, "End-to-end named entity and semantic concept extraction from speech," in *IEEE Spoken Language Technology Workshop*, Athens, Greece, Dec. 2018. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01987740

[5] P. Haghani, A. Narayanan, M. Bacchiani, G. Chuang, N. Gaur, P. Moreno, R. Prabhavalkar, Z. Qu, and A. Waters, "From audio to semantics: Approaches to end-to-end spoken language understanding," 2018.

[6] L. Lugosch, M. Ravanelli, P. Ignoto, V. S. Tomar, and Y. Bengio, "Speech model pre-training for end-to-end spoken language understanding," 2019.

[7] Y. Qian, R. Ubale, V. Ramanaryanan, P. Lange, D. Suendermann-Oeft, K. Evanini, and E. Tsuprun, "Exploring asr-free end-to-end modeling to improve spoken language understanding in a cloud-based dialog system," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Dec 2017, pp. 569–576.

[8] D. Serdyuk, Y. Wang, C. Fuegen, A. Kumar, B. Liu, and Y. Bengio, "Towards end-to-end spoken language understanding," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5754–5758, 2018.

[9] C. Gao, A. Rios-Navarro, X. Chen, T. Delbruck, and S.-C. Liu, "Edgedrnn: Enabling low-latency recurrent neural network edge inference," 2019.

[10] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," 2015.

[11] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018.

[12] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014.

[13] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *ICASSP*, 2016.

[14] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014.

[15] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014.

[16] J. Amoh and K. Odame, "An optimized recurrent unit for ultra-low-power keyword spotting," *IMWUT*, vol. 3, pp. 36:1–36:17, 2019.

[17] Y. Bengio, A. C. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, pp. 1798–1828, 2013.

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[19] O. Viikki and K. Laurila, "Cepstral domain segmental feature vector normalization for noise robust speech recognition," *Speech Communication*, vol. 25, no. 1, pp. 133 – 147, 1998.

[20] A. Zeyer, R. Schlüter, and H. Ney, "Towards online-recognition with deep bidirectional lstm acoustic models," in *INTERSPEECH*, 2016.

[21] T. N. Sainath, B. Kingsbury, A. Mohamed, and B. Ramabhadran, "Learning filter banks within a deep neural network framework," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, Dec 2013, pp. 297–302.

[22] J. Poncelet and H. V. hamme, "Multitask learning with capsule networks for speech-to-intent applications," 2020.

[23] E. Palogiannidi, I. Gkinis, G. Mastrapas, P. Mizera, and T. Stafylakis, "End-to-end architectures for asr-free spoken language understanding," 2019.